Using the Modular Open Systems Approach to Establish a More Rigorous Technical Baseline for Systems

System and Software Technology Conference
18-21 June 2007

Tampa Bay, FL

Cyrus Azani

DUSD (A&T)/SE/ED and NGC

1851 South Bell Street Suite 102

Arlington, VA 22202 USA

Agenda

- SE Revitalization
  - SE Challenges
  - Core Revitalization Strategy
- What is MOSA?
  - Why Open Systems?
  - Concept and definitions
  - MOSA Principles
  - Benefits vs. Challenges
- Developing Rigorous Technical Baselines Based on Open System Architecture
  - Technical Baselines and Architecture Definitions
  - Relationships (Architectures, Baselines, and Reviews)
  - Open Architecture Development Process
  - Implementation Challenges

SE Challenges

- Lack of uniform understanding of SE
- Lack of coherent SE policy
- Inadequate consideration of SE in program life cycle decisions
- Lack of alignment among multiple practitioner communities
- Lack of incentive or “forcing function” for execution of disciplined SE
- Multiple SE standards and models
- Evolutionary acquisition not well institutionalized
- Increasing system complexity
- Stovepipe system solutions

SE Revitalization

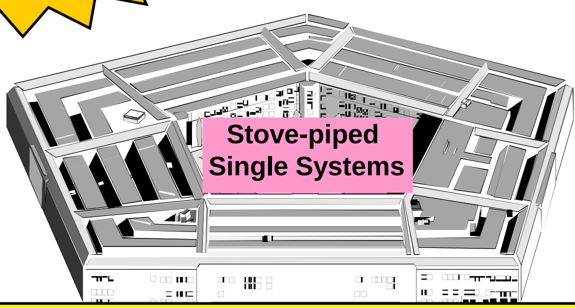
Core Strategy: Institutionalizing Systems Engineering across the Department by:

- **Raising Awareness of the Importance of Effective Systems Engineering and Promoting SE Application across the Defense Communities & Programs**
- **Establishing Succinct Policy, Procedures, Tools, Guidance, Education and Training**
- **Driving Technical Excellence into Programs via:**
  - *Proactive SE Management and Technical Planning*
  - *Rigorous Development of Technical Baselines*
  - Timely Insight into a Program's Technical Execution through *Event-based Technical Reviews*
- **Capturing and Sharing Best SE Practices**

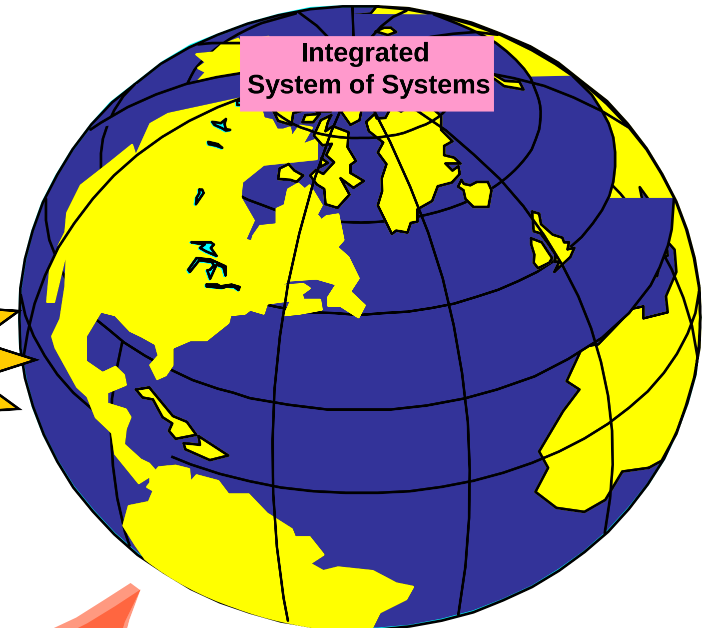
System Development Challenges……

Complex Integration of Incompatible Systems

1950s



21st Century

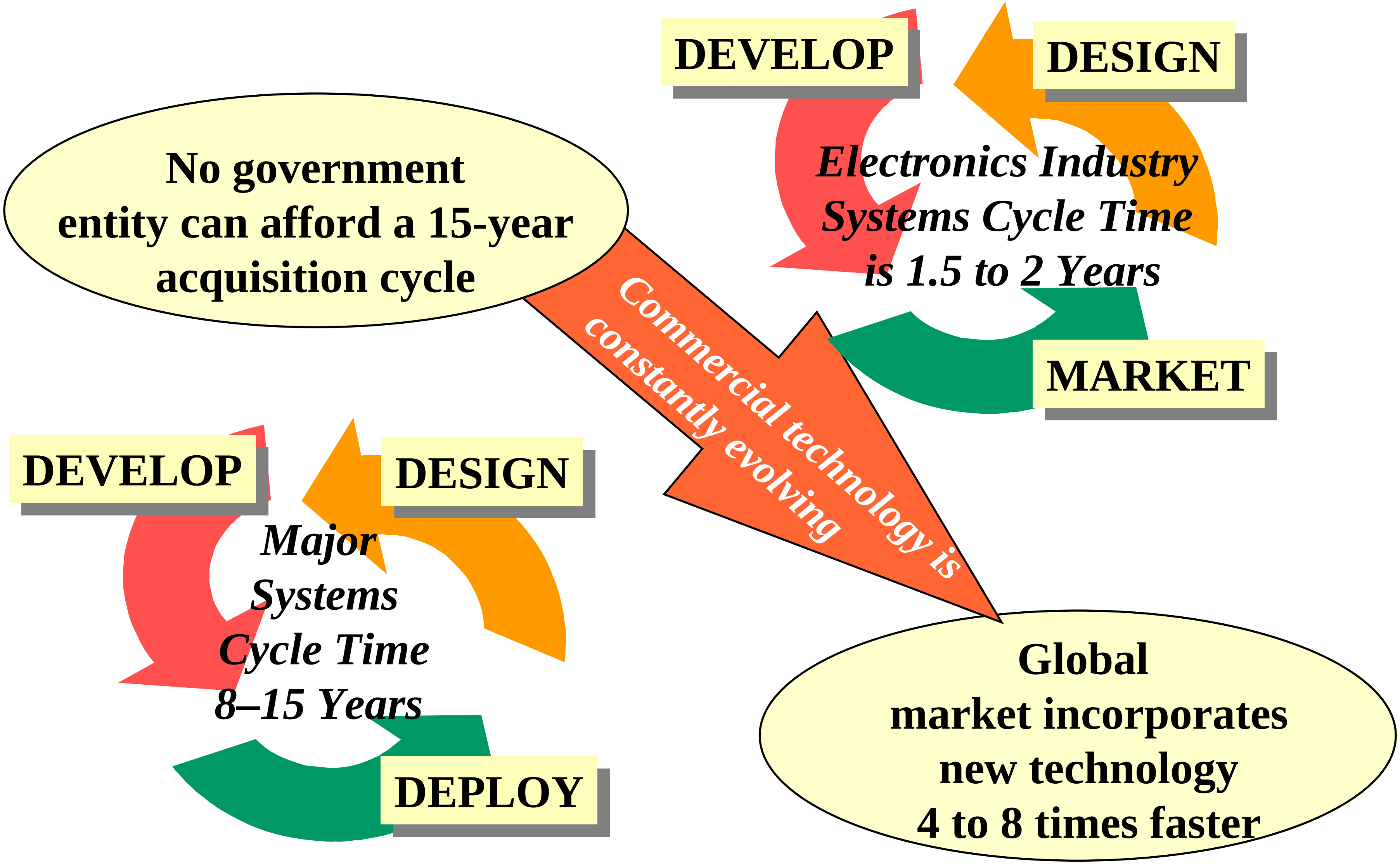


No single organization any longer "drives" development. Instead, it must use what others have developed

DEVELOPER & PRODUCER ➔ **BUYER & INTEGRATOR**

System Development Challenges….

Longer Military Systems Development Cycle

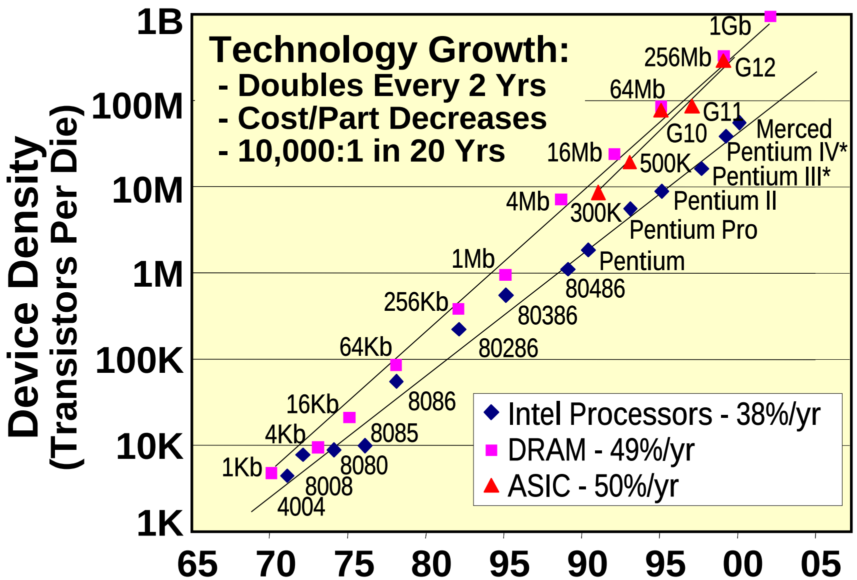


C. Azani

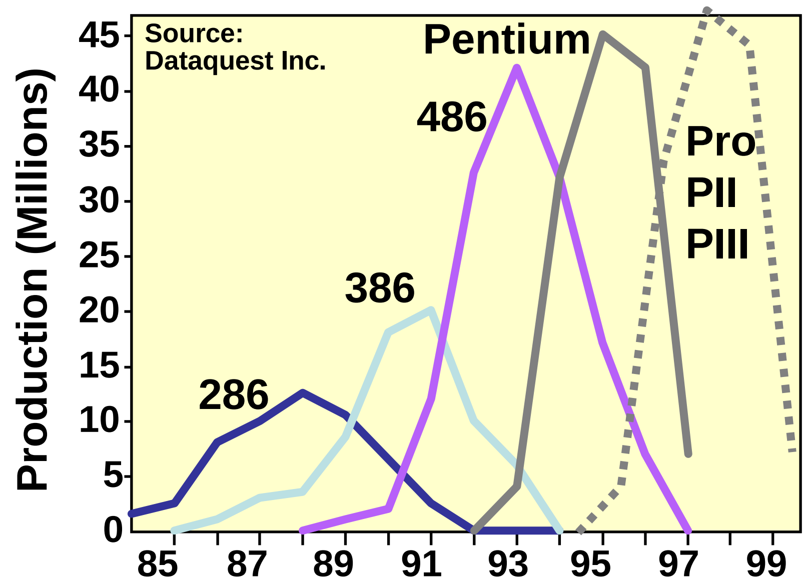
System Development Challenges….

New Technology Explosion and Shorter Commercial Product Life Time

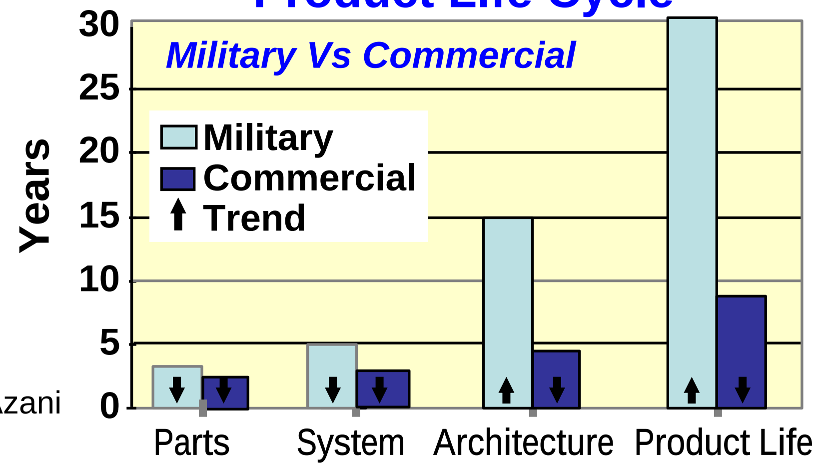
Technology Evolution



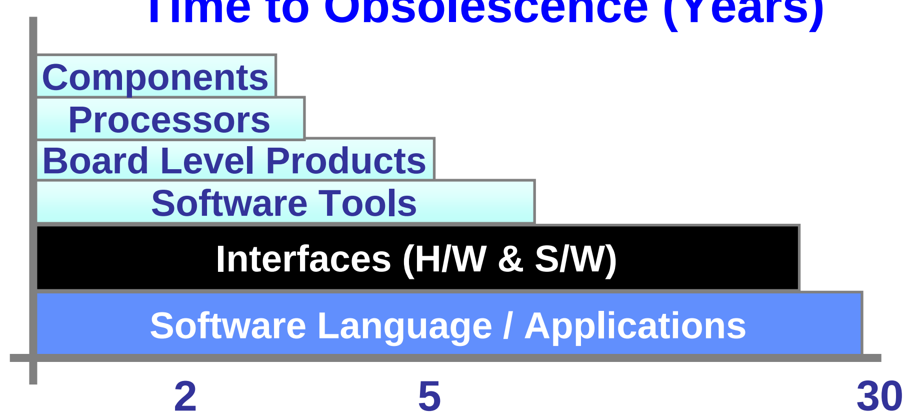
Shorter Product Lifetimes



Product Life Cycle



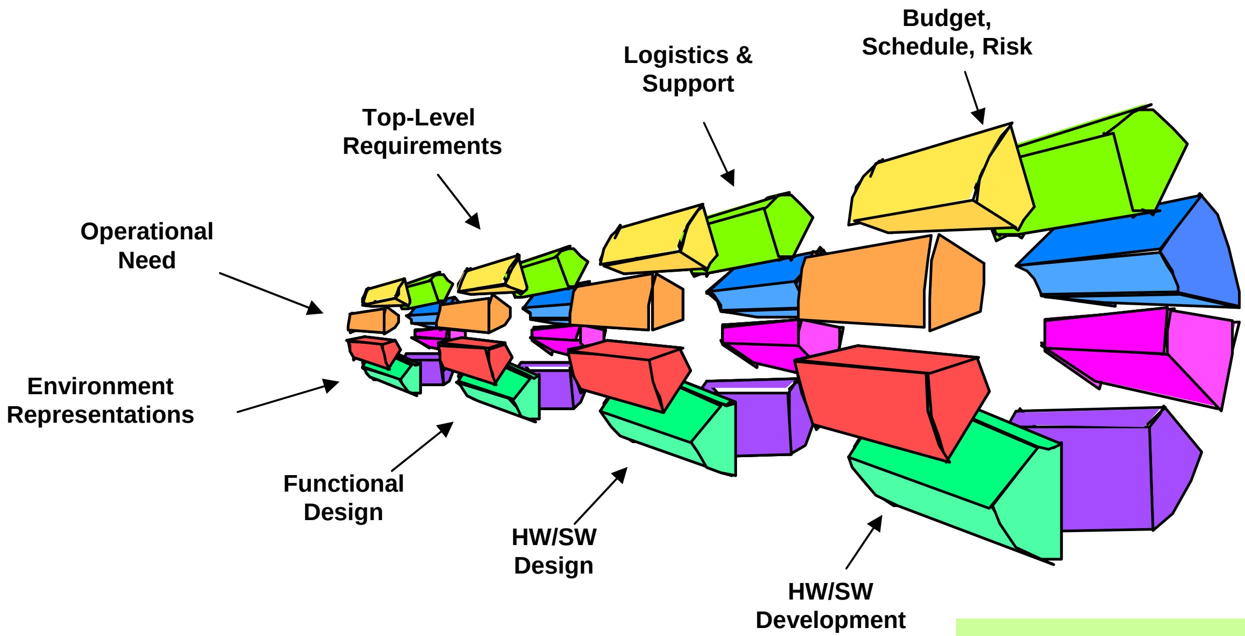
Time to Obsolescence (Years)



C. Azani

System Development Challenges….

Segregated Systems Engineering and Data Bases



<u>Challenges/Shortfalls</u>

Unique Stove-piped & Closed System Designs that:

- – Cost too much to develop
- – Cost too much to integrate
- – Cost too much to maintain
- – Cost too much to sustain
- – Cost too much to modernize
- – Take long time to develop
- – Do not follow modular design tenets
- – Won't talk to each other

C.

<u>Challenges/Shortfalls</u>

- – Inconsistent policies & procedures
- – Multiple stakeholders and info needs
- – Multiple data stores
- – Data redundancies and inconsistencies
- – Data trapped in proprietary tool formats
- – Limited interoperability among tools
- – Different meaning of data
- – Data translations are costly & error-prone
- – Non-standardized data format
- – Security issues

SE Challenges Can effectively be Met by Open Systems Design

- **What is an Open System?**

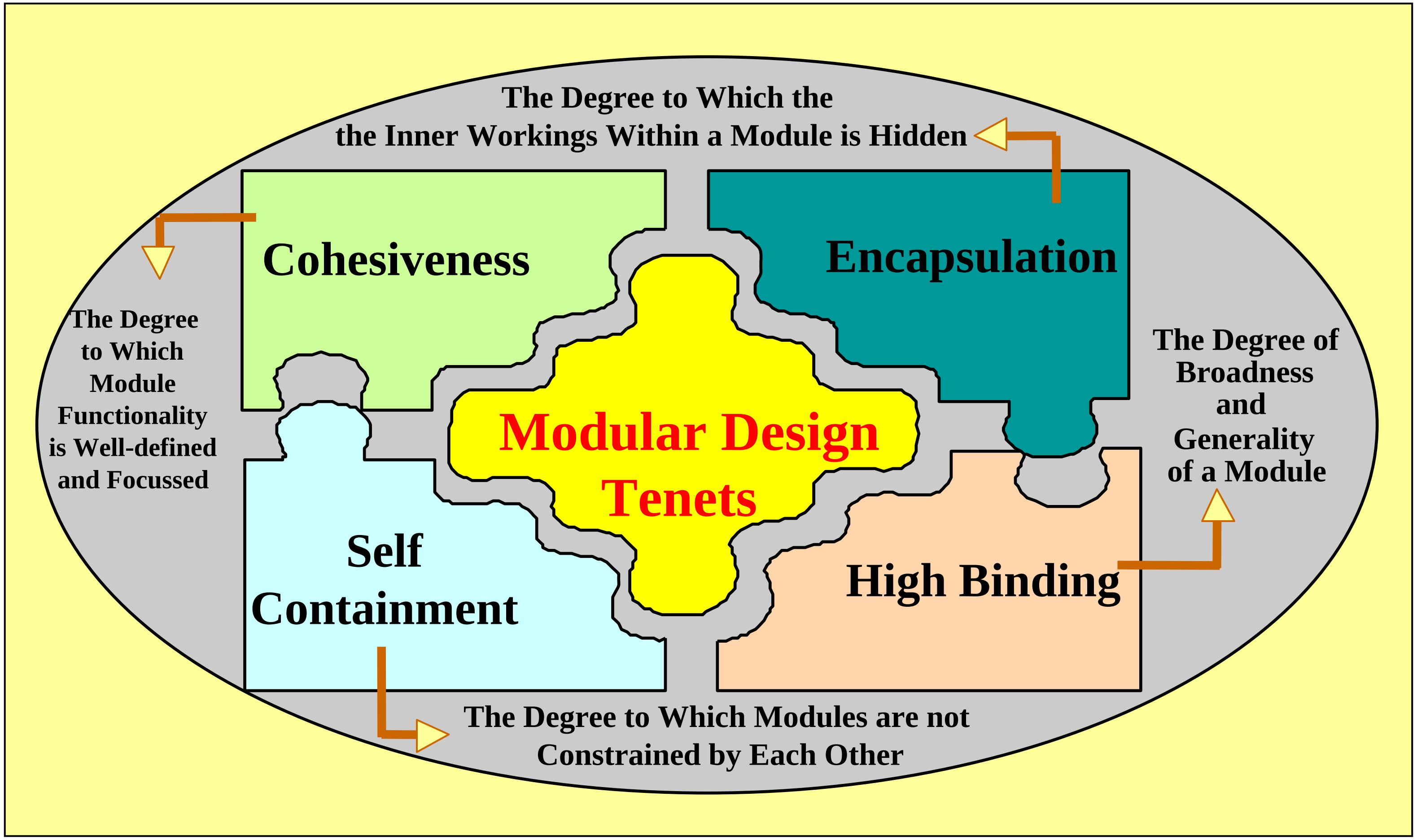
  A System that Exchanges:
  - **Material (e.g., hardware and software modules),**
  - **Energy (e.g., power and signals), and**
  - **Information (e.g., data exchange between two interoperable systems)**

  With its environment.

- **Based on DoD Definition, it is:**

  A System that:
  - **Employs modular design,**
  - **Uses widely supported and consensus based standards for its key interfaces, and**
  - **Is validated and verified to ensure the openness of its key interfaces.**

Modular Design Tenets

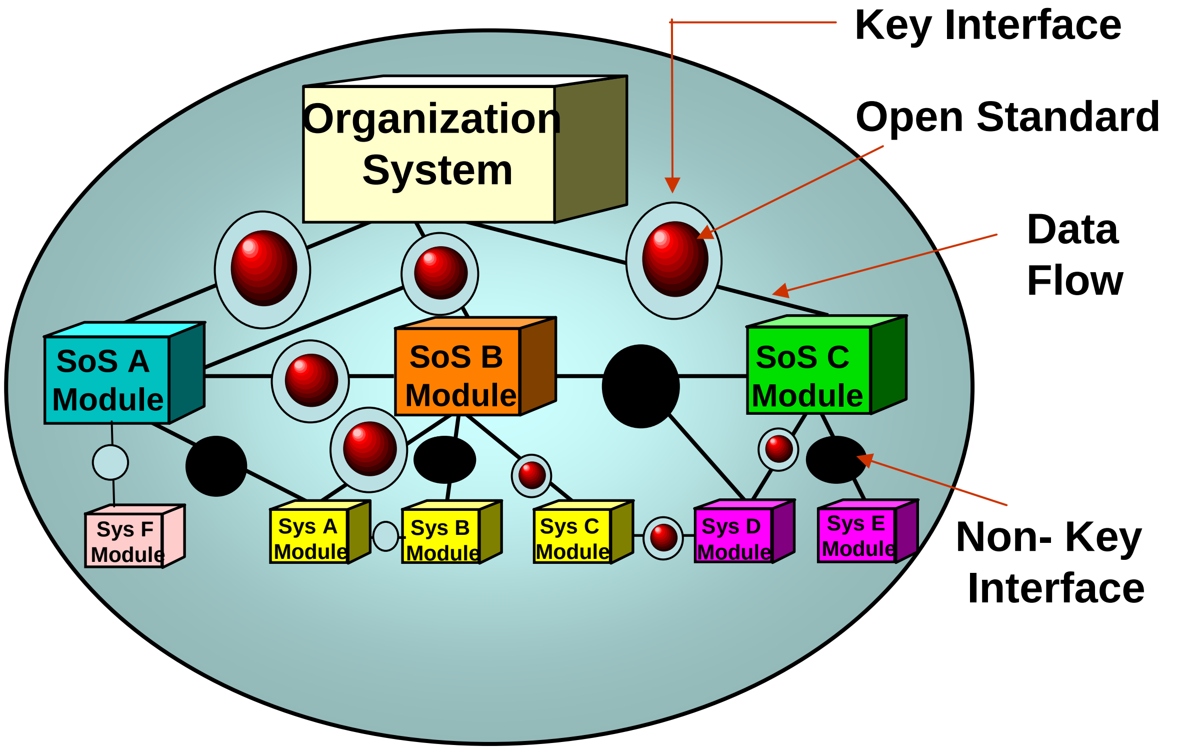


C. Azani

Interface Type and Characteristics

Types of Interfaces

- Mechanical (bolts, fasteners, connectors and plugs, etc.)
- Fluid (hydraulic, water, etc.)
- Environmental (thermal, nuclear (e.g., neutron, gamma, beta transmission rates and densities)
- Envelope (space allowances)
- Electrical (power, signals, etc.)
- Sequencing/Programming & timing
- Functional (data formats, etc.)



Interface Characteristics

- Key vs. non-key
- Functional vs. physical
- High vs. low performance
- Secure vs. non-secure
- Stable vs. changing
- Common vs. unique
- Standardized vs. non-standardized

Key Interface Designation Criteria:

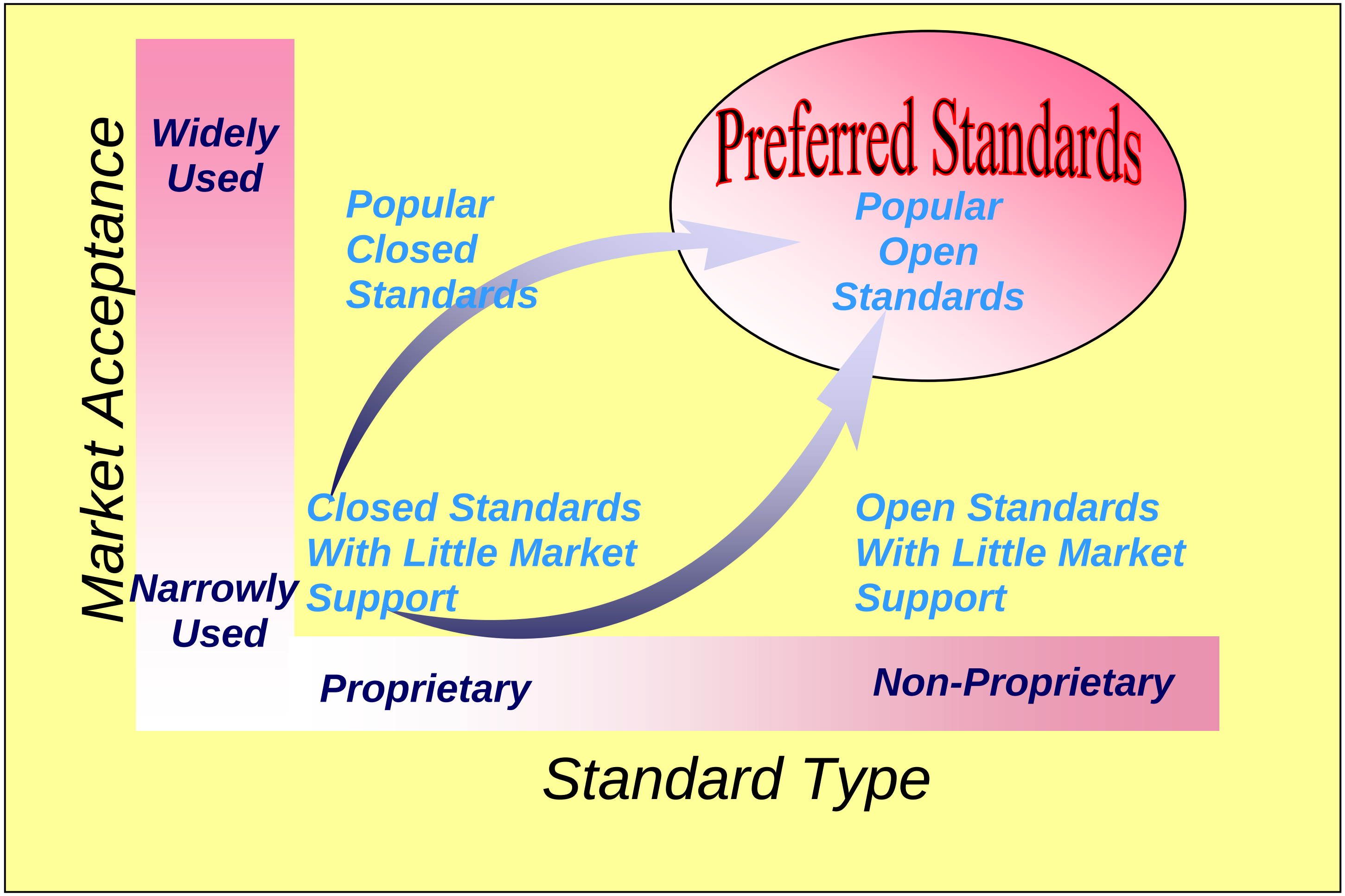
- High technology turnover rate
- Criticality of function
- Ease of integration
- Change frequency
- Interoperability
- Commonality/reuse
- High cost

Data Characteristics:

- Receive and transmit rules
- Data format, rate, volume, content, and meaning
- Signal frequency & timing
- Source & destination of data
- Processing, sharing & security

C. Azani

Open Standards



Standards that are widely used, consensus based, published and maintained by recognized industry standards organizations.

What is not Necessarily an Open Systems?

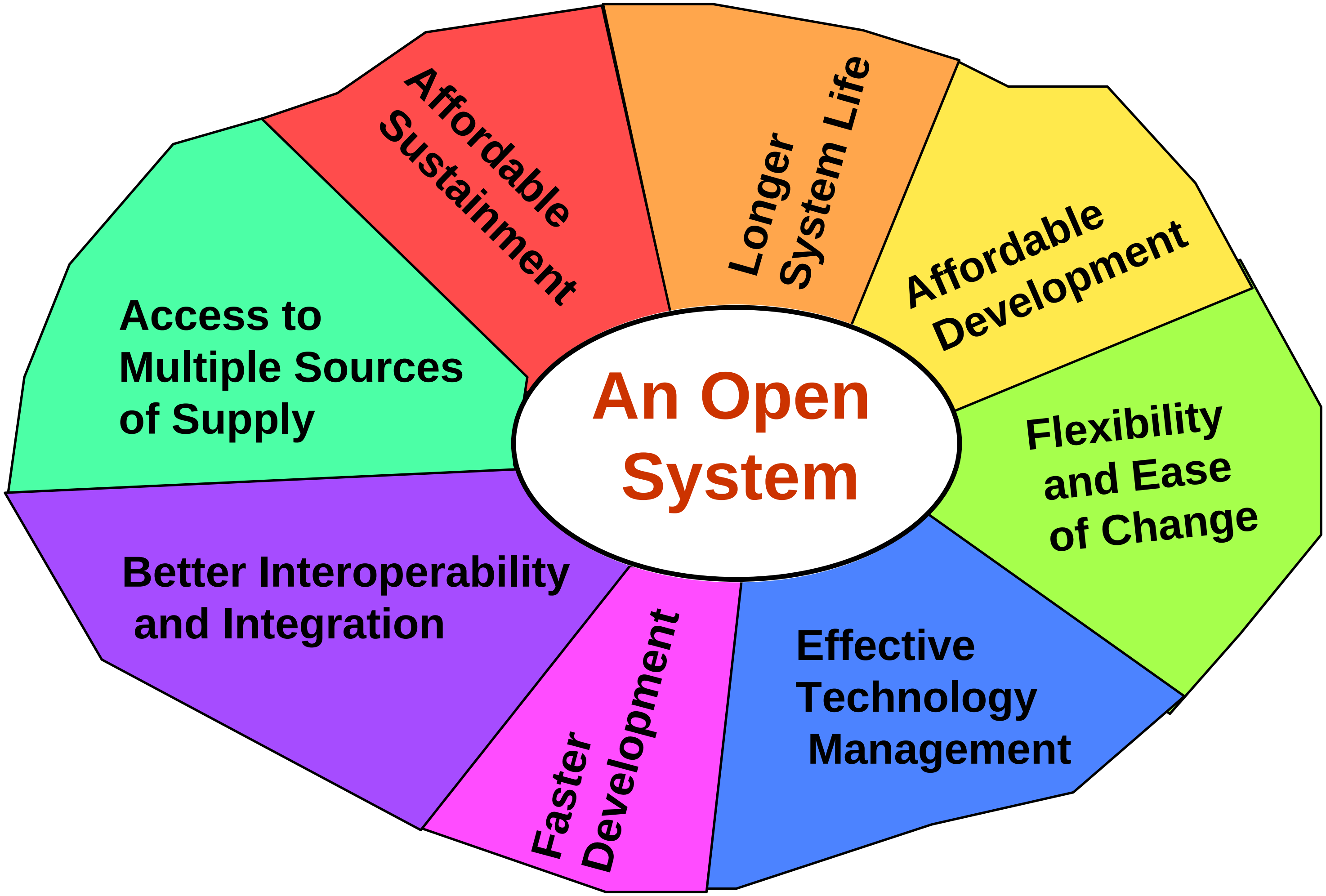
- Open Source Systems
- COTS Products
- F3I Systems
- Modular Systems

Are Not Necessarily $\neq$

OPEN SYSTEMS

Open System Design = **A design which is modular and based on non-proprietary interface standards that are broadly accepted and used throughout industry**

Open Systems Benefits



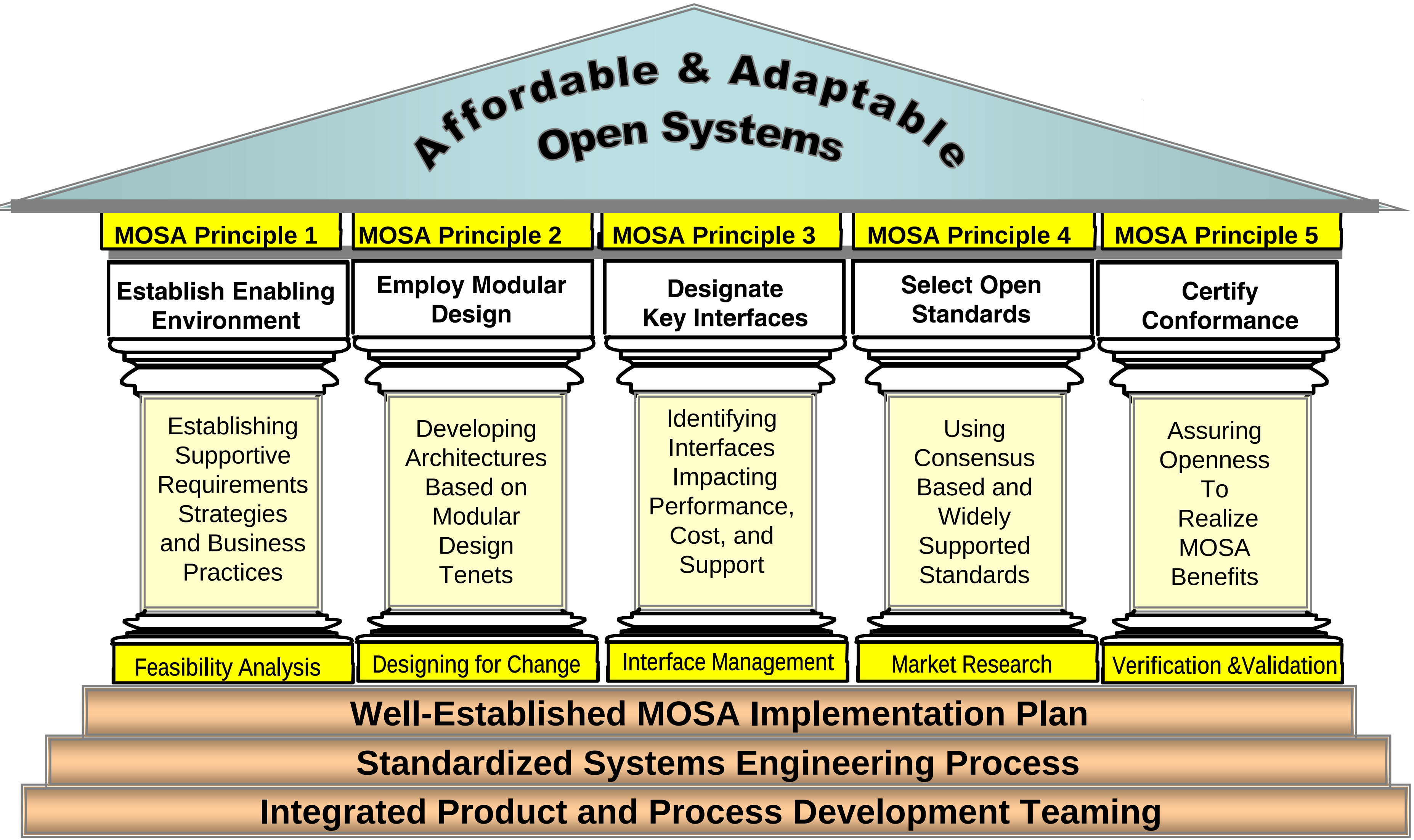
Like a Living Cell, an Open System has a Permeable Boundary to Effectively Exchange Information, Energy, and Material

MOSA Defined

An integrated business and technical strategy that :

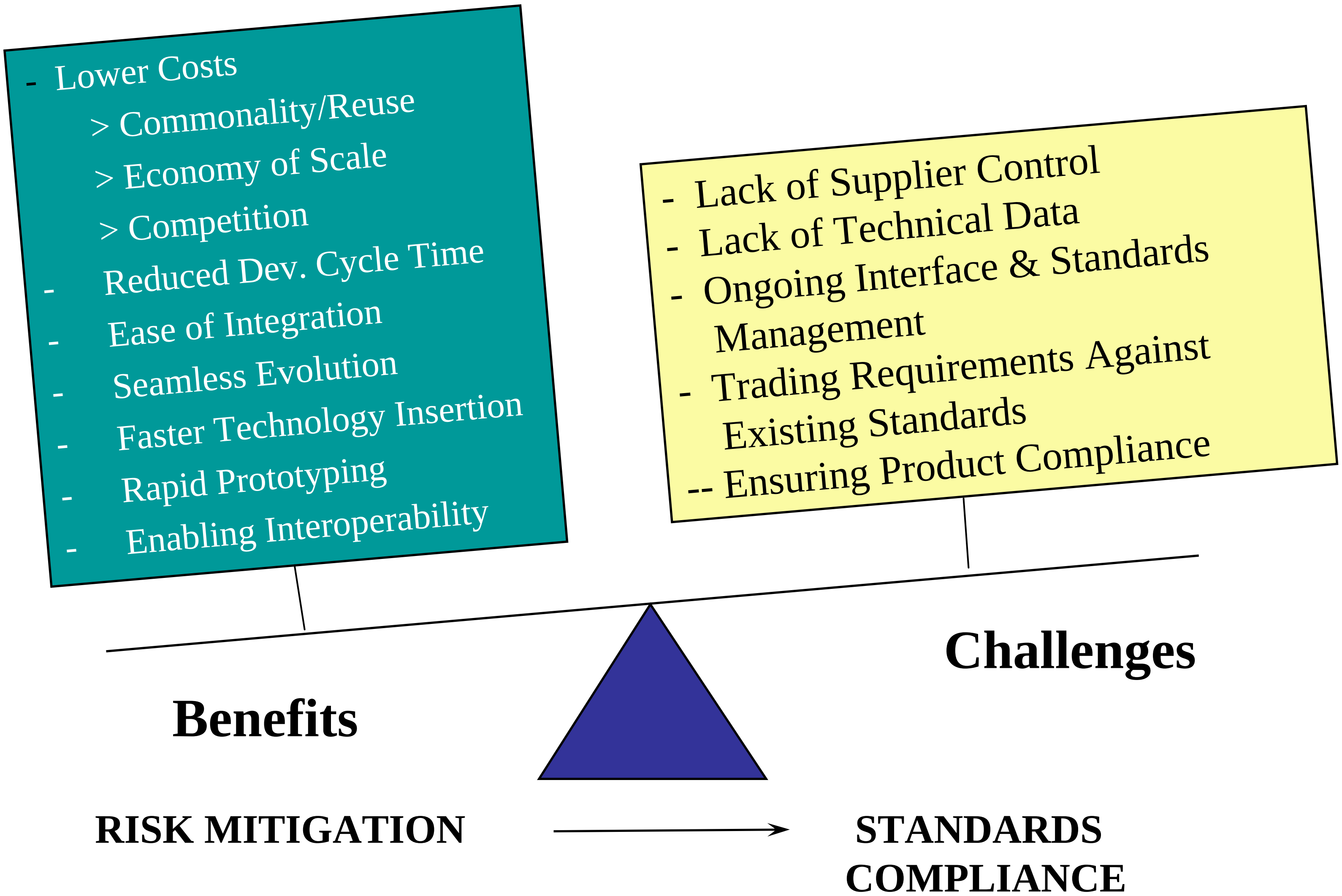
- **Provides an Enabling Environment for achieving:**
  - Affordable development & sustainment (both systems and SoS)
  - Evolutionary acquisition and improved capability
  - Effective interoperability and integration
- Employs **Modular design** and; where appropriate,
- Defines **Key Interfaces,**
- Using **Widely Supported, Consensus-based (i.e., open) Standards** that are published and maintained by a recognized industry standards organization; and
- Uses **Certified Conformant** products.

Principles for Effective MOSA Implementation



C. Azani

Open Systems Benefits vs. Challenges



C. Azani

Technical Baselines Defined

- Baseline - A group of formally accepted configuration items (CI) (subsystems, components, hardware and software products, deliverables, etc.) developed during a specific phase of the acquisition and development process.
- Technical Baseline - Describes technical characteristics of each configuration item at a particular time.
  - **Functional Baseline (system requirements baseline) -** the documentation describing a system's or top-level configuration item's functional and interface characteristics, design constraints, and the verification required to demonstrate the achievement of those specified characteristics.
  - **Allocated Baseline-** defines a subsystem's (CI's) functional and interface specifications that are allocated from those of the system or higher level CI, including design constraints and verification methods required to demonstrate that such specifications have been met.
  - **Product Baseline-** describes the end system product as built by the developers. It prescribes all necessary physical (including interface) characteristics of a configuration item during the production, operation, maintenance, and logistic support of its lifecycle.

Architecture Defined

- **Architecture -** The structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time (CJCSI 3170.01E).
- **System Architecture.** The arrangement of elements and subsystems and the allocation of functions to them to meet system requirements. INCOSE SE Handbook)
- **Functional Architecture –**
  - An arrangement of functions and their subfunctions and interfaces (internal and external) that defines the execution sequencing, conditions for control or data flow, and the performance requirements to satisfy the requirements baseline. (IEEE 1220)
  - The hierarchical arrangement of functions, their internal and external (external to the aggregation itself) functional interfaces and external physical interfaces, their respective functional and performance requirements, and the design constraints. (INCOSE SE Handbook)
- **Physical Architecture -** The hierarchical arrangement of product and process solutions, their functional and performance requirements; their internal and external (external to the aggregation itself) functional and physical interfaces and requirements, and the physical constraints that form the basis of design requirements. INCOSE SE Handbook)
- **Open Architecture -** An architecture that employs open standards for key interfaces within a system.

The Many Faces of Architecture

- "...explicit ways to depict what you are building so multiple people can have a common understanding to coordinate their activities." *John Zachman*

- "Even if we could document the business strategy...and determine how to relate that business strategy to an I.S. strategy...a fundamental problem remained: how to get from those strategy matrices to implementation...During the **1980s, I** became convinced that architecture, whatever that was, was the thing that *bridged the strategy and its implementation." John Zachman*

- "In the earliest stage of a project it [architecting] is a *structuring* of an *unstructured* mix of dreams, hopes, needs, and technical possibilities when what is most needed has been called an inspired *synthesizing of feasible technologies." Rechtin & Maier*

- "...the architect's basic role is the *reconciliation of a physical form with* the client's need for *function, cost,* certification, and *technical feasibility." Rechtin & Maier*

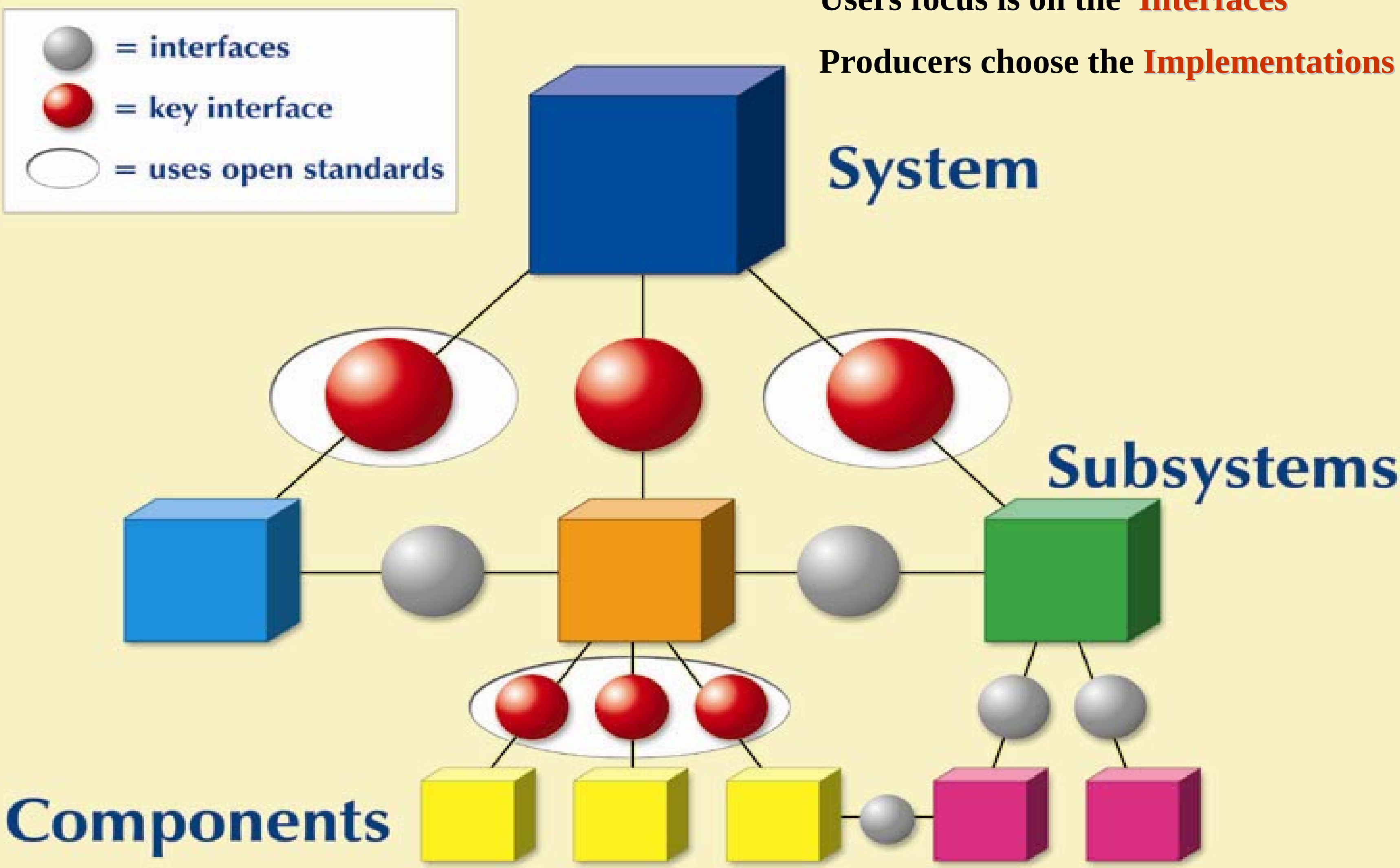
The Many Faces of Architecture continued

- "Architecture...reduces *complexity…It* enables *management* of *unpredictability* and *change...permits* many systems and organizations to be *developed independently* and still work together." *Morris & Ferguson*
- "Technology can be so vast and overwhelming; architecture puts a *framework around technology." Emillie Schmidt*
- "Fundamentally, an *information architecture* is a political doctrine that specifies who will have what types of information to make decisions." *Paul Strassmann*
- "...architecture has other 'audiences' than just developers - it serves as *basis for analysis* and *decision-making* throughout the life cycle...must be *usable* by *end users, acquirers,* the *system owner* and operator…be able to *support technical,* cost and programmatic decisions." *Emery, Hilliard II, & Rice*
- "The *architectural* design of large systems also has *consequences* for how you define the *development process* and *organization." VanDerLinden &Muller*

Open Architectures Focus on Interfaces

Users focus is on the Interfaces

Producers choose the Implementations



An Architecture Development Process

- ***Determine what will be represented*** What types of tasks/objects/functions must we include?
- ***Determine required task/object attributes***

  What do we need to know about each one?
- ***Determine required interrelationships*** How do they relate to each other?
- ***Determine environment rules***

  What rules govern relationships & future evolution?

  What standards shall be used?
- ***Determine representation media***

  What are the best ways to display each object and relationship?
- **Construct representative views**

  What capability do we need to observe from different perspectives and simultaneously look at different system views

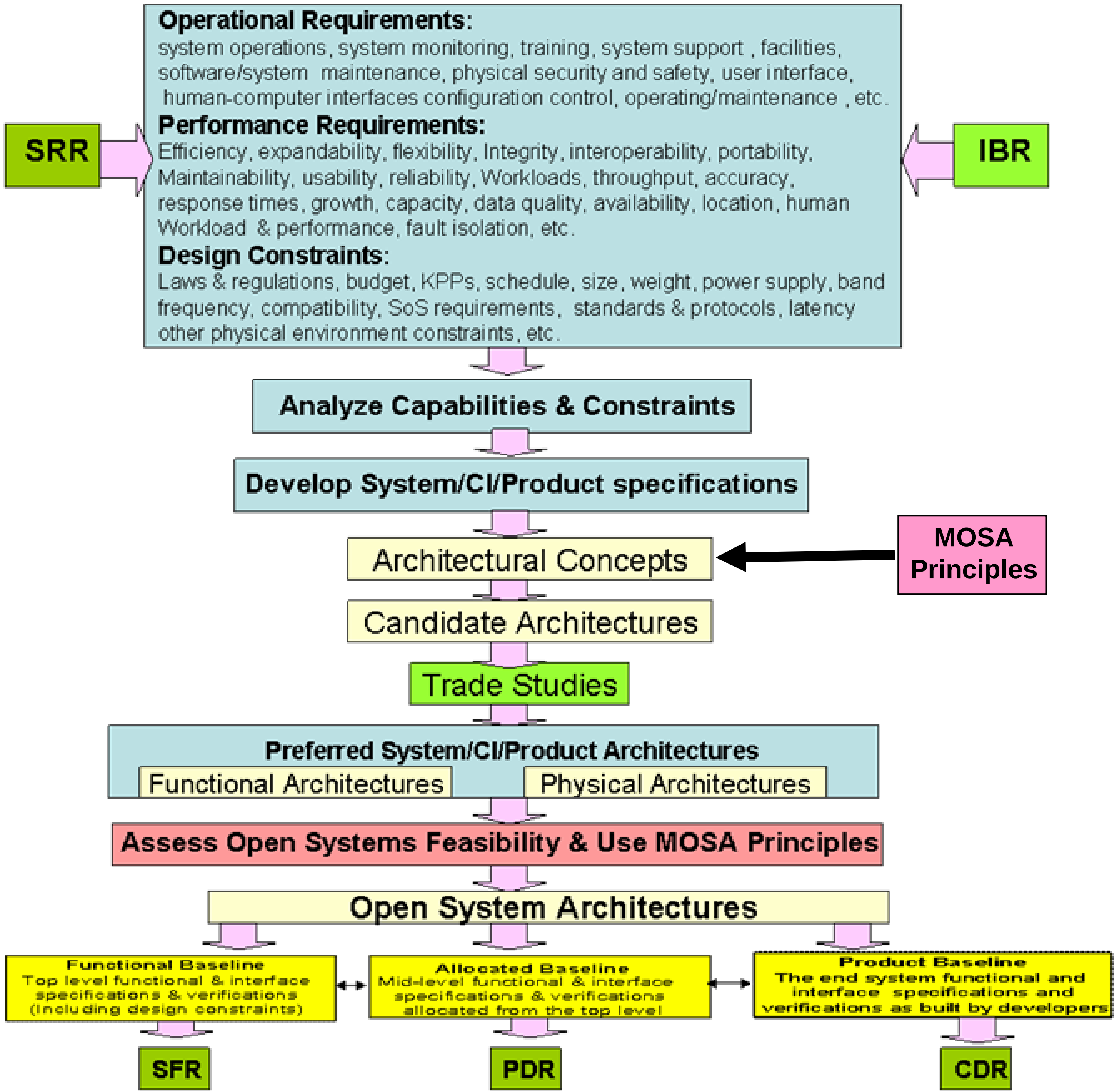
- ***Determine data requirements***

  What do we need to know about the relationships?

  What are key data sources and how could we build capability to share data?
- ***Determine initial query requirements--***What queries do we know we'll need to feed required fields and reports?
- **Analyze and implement**

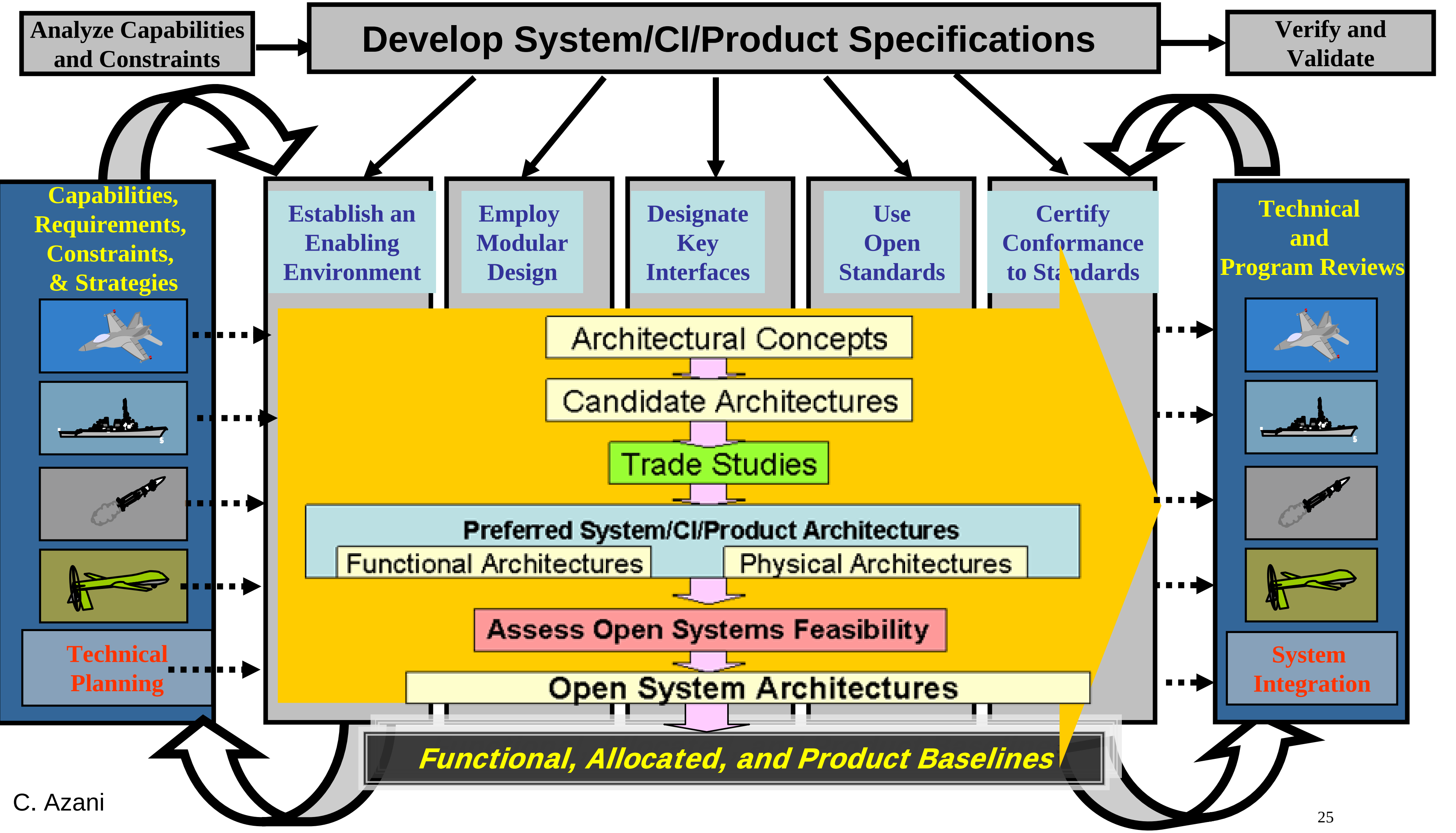
  What are the trade offs and what areas are missing and why?
- **Finalize & document the selected architecture**

An Open Architecture Development Process



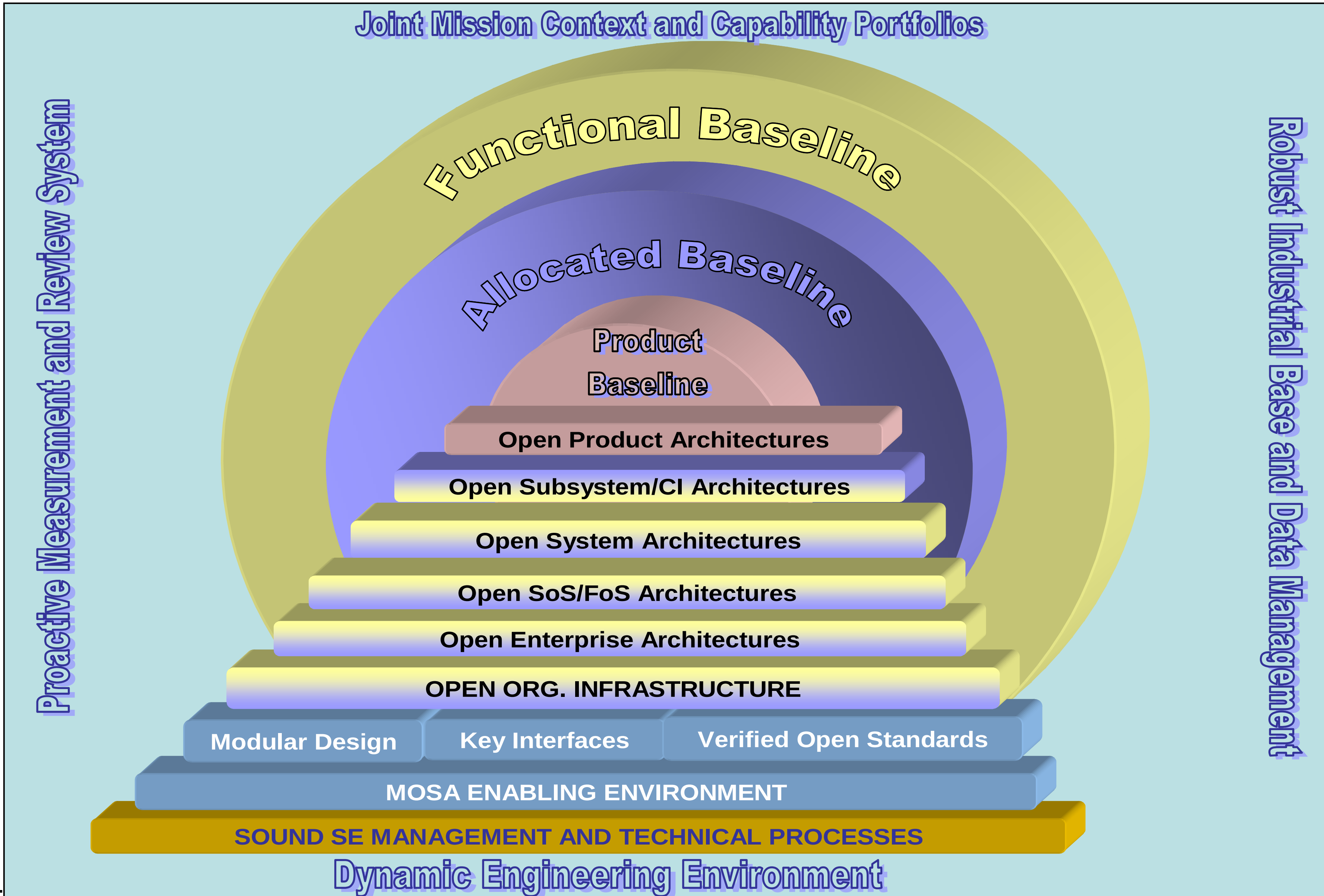
C. Azani

An Open Architecture Development Process



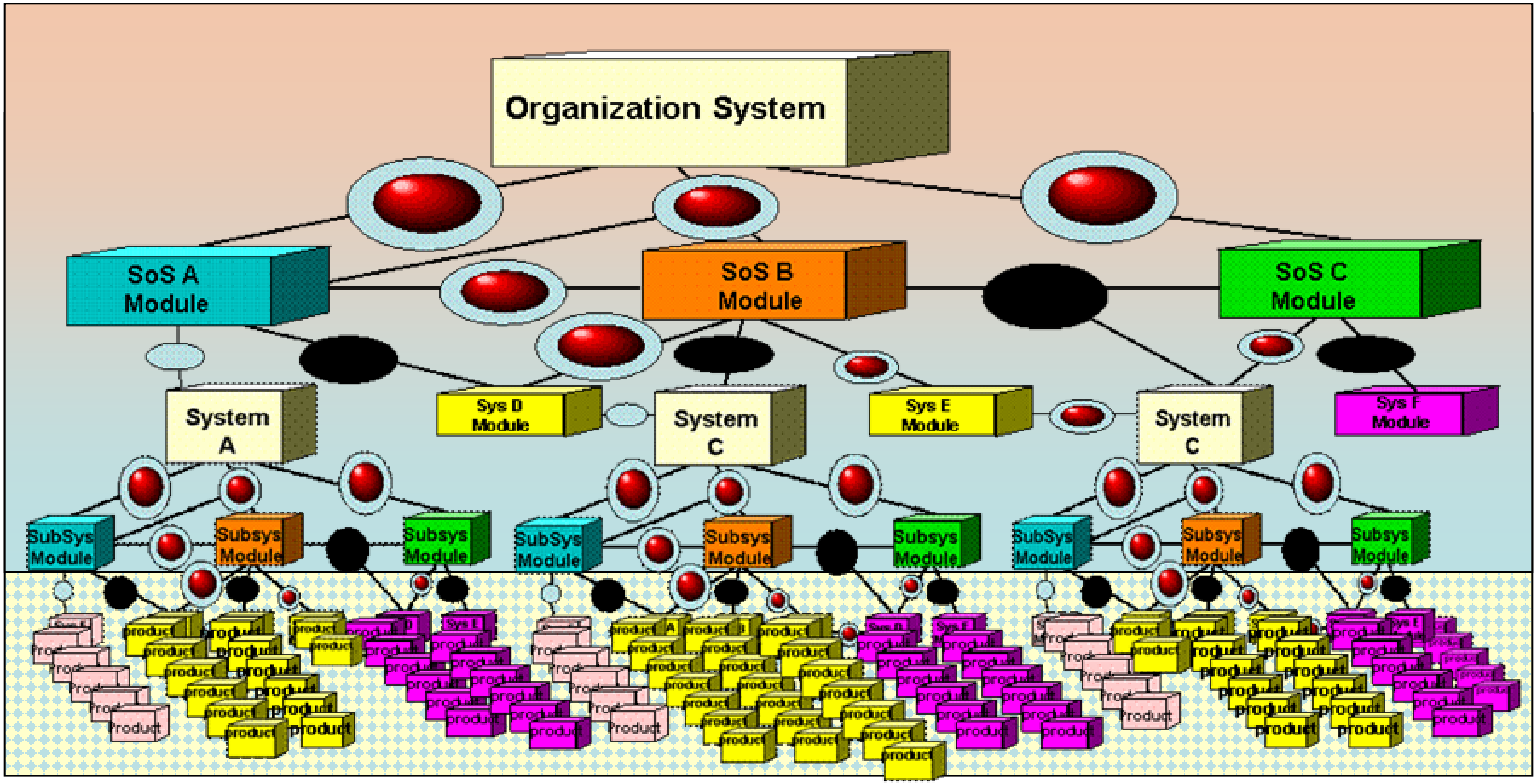
C. Azani

Hierarchy of Open Architectures



C.

Hierarchy of Open Architectures

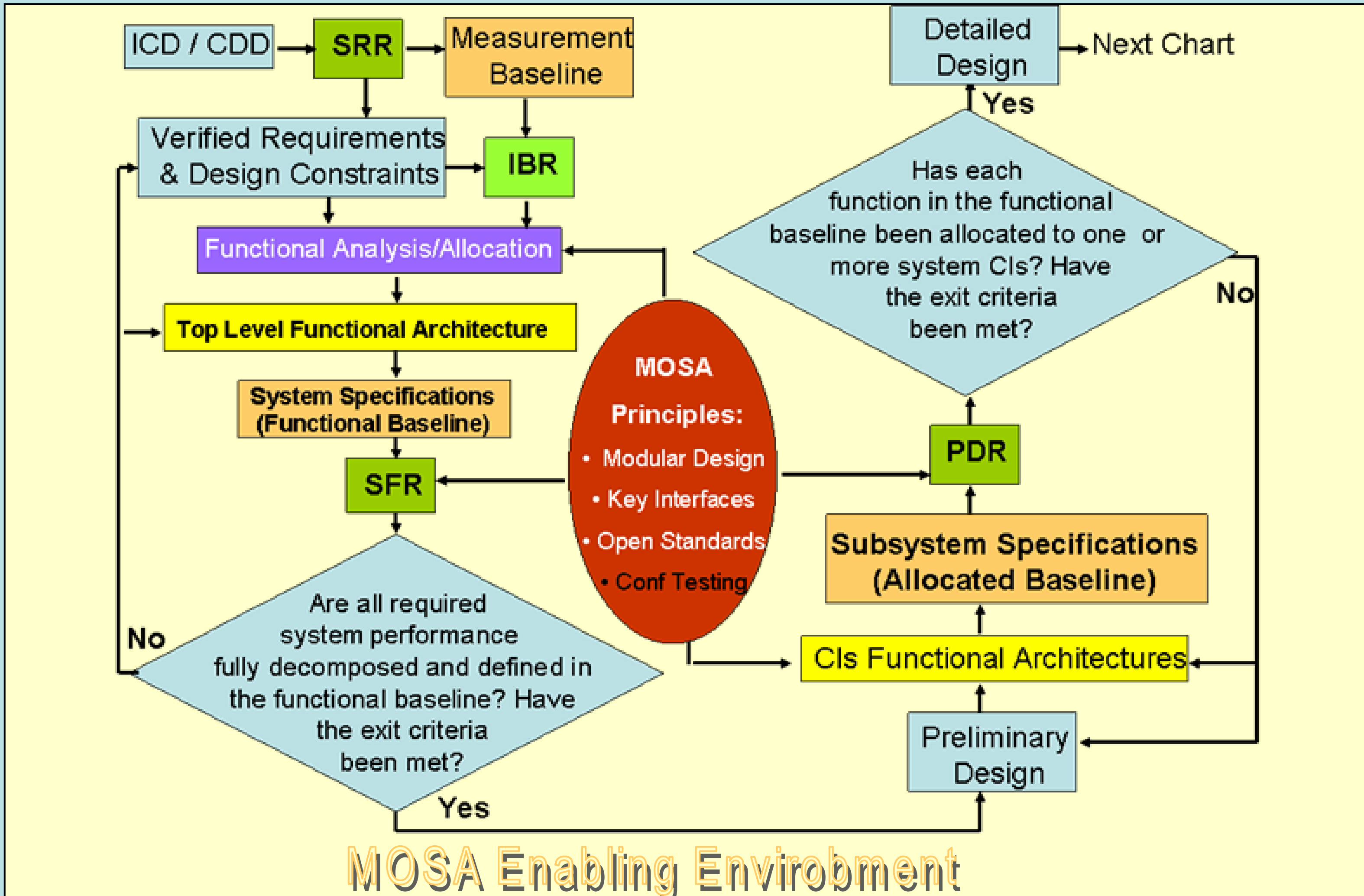


Top level Specifications

Allocated Specifications

C. Product Specifications

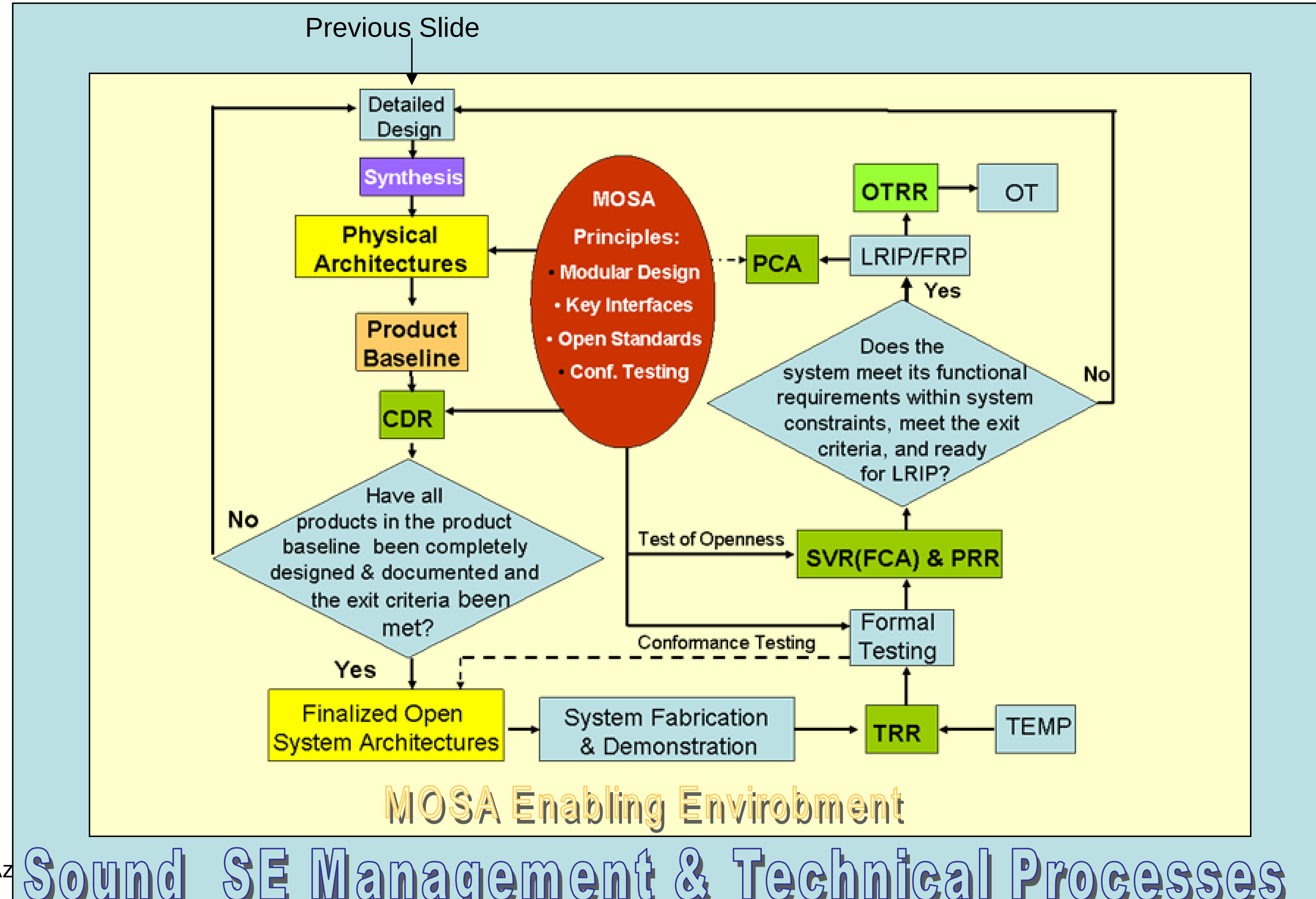
Architectures Relationship with Technical Baseline & Reviews



C. Az

Sound SE Management & Technical Processes

Architectures Relationship with Technical Baseline & Reviews



C. Az

10 Steps Methodology for Developing and Integrating Open Architectures with Technical Baselines

1. Decompose the system into functions and employ modular design tenets to group system functions into self-contained, cohesive, encapsulated, and decoupled CIs
2. Arrange functional modules (CIs) and their interfaces into architectural concepts
3. Narrow down the concepts to a few candidate architectures based on programmatic, technical and total life cycle concerns/criteria
4. Conduct trade studies, supportability, and cost analyses on candidate architectural concepts to select a preferred architecture for the system that balances cost effectiveness, performance, schedule, supportability; and robustness.
5. Refine the functional and interface specifications (including design constraints and verification methods) for the selected system architecture based on MOSA principles
6. Decompose the system architecture into CI architectures and allocate functional and interface specifications from those of the higher level systems (including design constraints and verification methods) to these CIs.
7. Reiterate (configure and reconfigure the architectures) as needed to develop networks of modular, secured, and open interoperable architectures for all CIs, systems, or SoS.
8. Document functional, allocated, and product baselines including profiles of open standards for all key system interfaces
9. Ensure conformance/compliance to NR-KPP, open standards, and interoperability and other policies/rules
10. Manage key interfaces via collaborative joint/coalition change management councils

Conclusion: Open architectures

- Facilitate development of technical baselines (e.g., facilitate functional partitioning and allocation via modular design)
- Reduce the risks associated with integration of incompatible systems, CIs, and product specifications
- Enable more robust management of interfaces within each baseline
- Enable integration of new and legacy systems into networks of interoperable system of systems
- Enable fast and affordable reconfiguration of systems
- Provide access to global competitive markets for products and tools
- Facilitate commonality of hardware, software, and tools
- Make the upgrade and evolution of systems/CIs/products more affordable by leveraging commercial market place investment in new technology and products

Robust Technical Baselines are Enabled through Development of Standards-based Networks of Modular and Reconfigurable Open Architectures

Questions……..